Demonstration of a Time-predictable Flight Controller on a Multicore Processor

Oktay Barış*, Shibarchi Majumder†, Tórur Biskopstø Strøm*, Anders La Cour-Harbo†,
Jens Sparsø* , Thomas Bak†, and Martin Schoeberl*
*Technical University of Denmark, †Aalborg University

***Abstract*—Unmanned aerial vehicles, or drones, have drawn extensive attention during the last decade together with the maturity of the technology. Often, real-time requirements are needed when they are deployed for critical missions. The increasing computational demand for drones leads to a shift towards multicore architectures. However, the timing analysis of multicore systems is a challenging task due to the timing interference between processor cores. In this demonstration, we deploy a parallelized flight controller system on a multicore architecture. We provide timing analysis of the system, and test it with a processor-in-the-loop setup, which includes a flight simulator connected to the flight controller running on a time-predictable multicore platform.**

***Index Terms*—real-time systems, flight controller, multicore.**

I. Introduction

Drones are being deployed in numerous projects, and their usage is expected to soon grow considerably [4]. The proliferation of multicore architectures has paved the way for replacing radio-controlled navigation with on-board real-time tasks that adapt the flight path.

Typically, an autonomous drone operates by reading inertial sensors to predict its current attitude,[1] after which a controller updates actuators. Thus, it can carry out mission objectives without any remote-control.

Real-time tasks may be mapped to multiple dedicated cores on the same chip and cooperate with each other via inter-core communication mechanisms. Timing analysis of real-time systems deployed on multicore architectures is considered a challenging task due to timing interference, introduced by concurrently running tasks on different processor cores [1].

This demonstration presents a time-predictable multicore architecture for a flight controller. Figure 1 shows the setup of the demonstrator. The control loop runs on a dedicated processor core, while a parallelized version of the state estimator algorithm is off-loaded onto two different cores with a fork-and-join operation when the estimator function is called.

We use the T-CREST multicore processor [6], [8]. The T-CREST platform contains several Patmos [9] cores connected to several communication options: (1) a shared, external memory with a time-predictable memory arbiter [7]; (2) a shared scratchpad memory [10]; and (3) to the Argo network-on-chip [3].

[1]Attitude is an avionics term, meaning the orientation of the drone relative to the horizon.

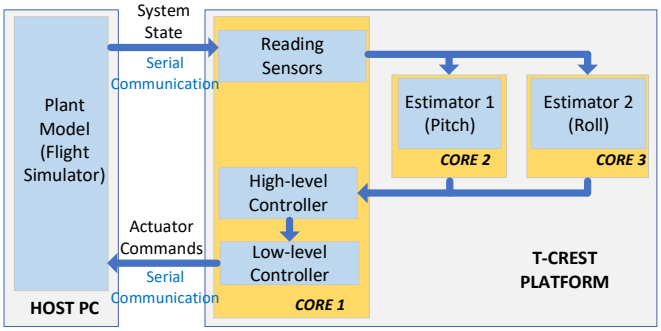

Fig. 1. System overview: The estimator is parallelized and mapped to two different cores, and connected to a flight simulator for processor-in-the-loop testing.

There is a lack of literature on fully analyzable multicore flight control systems. In recent work, Cheng et al. [2] introduce an end-to-end timing analysis for a drone flight control system considering a set of tasks in a communication pipeline, each mapped to a separate virtual processor that runs on a single physical processor.

In another work, Rocha et al. [5] port prototype avionics applications to the time-predictable T-CREST multicore processor. The applications are based on the Integrated Modular Avionics concept, which goal is to reduce hardware by sharing resources without any interference among applications. However, in this paper we aim to port traditionally sequential flight control systems to multicores.

The contribution of this demonstration is a parallelized time-predictable flight controller deployed on a multicore architecture.

II. The Flight Controller

Drones are equipped with sensors that provide information about the current attitude of the drone. One of the most common sensors is an inertial measurement unit, which measures accelerations, rotation rates, and the magnetic field of the earth to determine the drone orientation.

The flight controller periodically, (a) reads the inertial measurement unit sensor, (b) integrates (fuses) data coming from the sensors with a state estimator algorithm, (c) computes the control law, and (d) updates the actuators to maintain the body balance.

The flight controller is implemented as cascaded PID controllers, which consist of a high-level controller and a set

TABLE I
RESULTS FROM WORST-CASE EXECUTION TIME ANALYSIS FOR SINGLE CORE

Function	Clock Cycles
Estimator (pitch and roll)	486,398
High- and low-level controllers	17,189
Control loop	532,399

of low-level controllers. The high-level controller mimics pilot/user input, thus implementing predefined flight-modes. The low-level controllers include a longitudinal controller (pitch control) and a lateral controller (roll control). The low-level controllers update actuators to keep the reference attitude according to high-level controller commands.

An estimator algorithm, a Kalman filter, is used for sensor data fusion for extracting otherwise unmeasurable variables from the system, as the measurement data is highly uncertain. The estimator fuses pitch and roll degrees to be used by the longitudinal and lateral controllers, respectively. When the estimator is parallelized, two estimator instances deployed on two different cores predict the pitch and roll degrees separately.

III. Evaluation and Demonstration

For the demonstration, we use a processor-in-the-loop setup, as illustrated in Figure 1. In the setup, the flight controller is deployed on a dedicated T-CREST multicore platform interacting with a flight simulator, which runs on a Windows computer. The flight controller and the host computer communicate over a serial line. We use a commercially available flight simulator (XPlane 11). The flight simulator exchanges data in the form of UDP packets, It provides the requested sensor data as an output, and obtains the actuator update values from the flight controller as an input. An interfacing program is used to convert the UDP packets into a serial communication data for interfacing the simulator with the embedded flight controller running on the T-CREST platform. The sensor data is read and sent at 20 Hz, which is also the frequency of the control loop.

Table I provides worst-case execution time analysis results of a traditional single core flight control system. Here, the estimator algorithm becomes a bottleneck for the control loop by taking nearly 91% of its execution time. Therefore, the estimator algorithm is parallelized to run on two different cores fusing pitch and roll values separately. In the T-CREST platform, however, access to the main memory is TDM arbitrated, and increasing the number of processor cores also increases the memory access cost for the sake of time predictability. When external memory is used, this nature of the platform prevents a possible speed up that can be obtained by parallelizing the application.

The source code for the flight controller is available in open source at: https://github.com/predict-drone/flight-control.

IV. Future Work

As a future work, we will extend this work to cover inter-core communication concepts including exploring on-chip shared memories and a network-on-chip. We aim to compare the different inter-core communication concepts. Additionally, we will further explore parallelizable estimator algorithms for flight control systems.

V. Conclusion

In this demonstration, a flight controller system is parallelized and deployed on a time-predictable multicore processor platform. The flight controller is evaluated with a processor-in-the-loop testing setup. In this setup, we use a flight simulator running on a host computer connected to the controller. Timing analysis of the system shows that the estimator dominates the execution time and therefore is the candidate for parallelization.

Acknowledgement

The work was partially funded by the Danish Council for Independent Research | Technology and Production Sciences under the project PREDICT (http://predict.compute.dtu.dk/), contract no. 4184-00127A.

References

[1] Pierre Bieber, Frédéric Boniol, Marc Boyer, Eric Noulard, and Claire Pagetti. New challenges for future avionic architectures. *AerospaceLab*, (4):p–1, 2012.

[2] Zhuoqun Cheng, Richard West, and Craig Einstein. End-to-end analysis and design of a drone flight controller. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 37(11):2404–2415, 2018.

[3] Evangelia Kasapaki, Martin Schoeberl, Rasmus Bo Sørensen, Christian T. Müller, Kees Goossens, and Jens Sparsø. Argo: A real-time network-on-chip architecture with an efficient GALS implementation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24:479–492, 2016.

[4] Naser Hossein Motlagh, Tarik Taleb, and Osama Arouk. Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives. *IEEE Internet of Things Journal*, 3(6):899–922, 2016.

[5] André Rocha, Cláudio Silva, Rasmus Bo Sørensen, Jens Sparsø, and Martin Schoeberl. Avionics applications on a time-predictable chip-multiprocessor. In *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2016)*, pages 777–785. IEEE Computer Society, Feb 2016.

[6] Martin Schoeberl, Sahar Abbaspour, Benny Akesson, Neil Audsley, Raffaele Capasso, Jamie Garside, Kees Goossens, Sven Goossens, Scott Hansen, Reinhold Heckmann, Stefan Hepp, Benedikt Huber, Alexander Jordan, Evangelia Kasapaki, Jens Knoop, Yonghui Li, Daniel Prokesch, Wolfgang Puffitsch, Peter Puschner, André Rocha, Cláudio Silva, Jens Sparsø, and Alessandro Tocchi. T-CREST: Time-predictable multi-core architecture for embedded systems. *Journal of Systems Architecture*, 61(9):449–471, 2015.

[7] Martin Schoeberl, David VH Chong, Wolfgang Puffitsch, and Jens Sparsø. A time-predictable memory network-on-chip. In *Proceedings of the 14th International Workshop on Worst-Case Execution Time Analysis (WCET 2014)*, pages 53–62, Madrid, Spain, July 2014.

[8] Martin Schoeberl, Luca Pezzarossa, and Jens Sparsø. A multicore processor for time-critical applications. *IEEE Design Test*, 35:38–47, 2018.

[9] Martin Schoeberl, Wolfgang Puffitsch, Stefan Hepp, Benedikt Huber, and Daniel Prokesch. Patmos: A time-predictable microprocessor. *Real-Time Systems*, 54(2):389–423, Apr 2018.

[10] Martin Schoeberl, Tórur Biskopstø Strøm, Oktay Baris, and Jens Sparsø. Scratchpad memories with ownership. In *2019 Design, Automation and Test in Europe Conference Exhibition (DATE)*, 2019.